May 2002

object oriented programming

ترجمة تواتي عمر

مراجعة أسماء

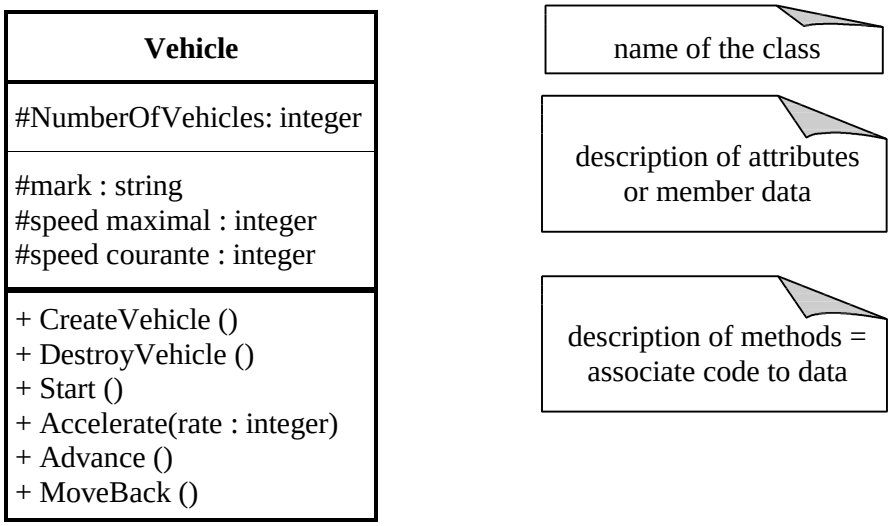
مدخل إلى الكائنات

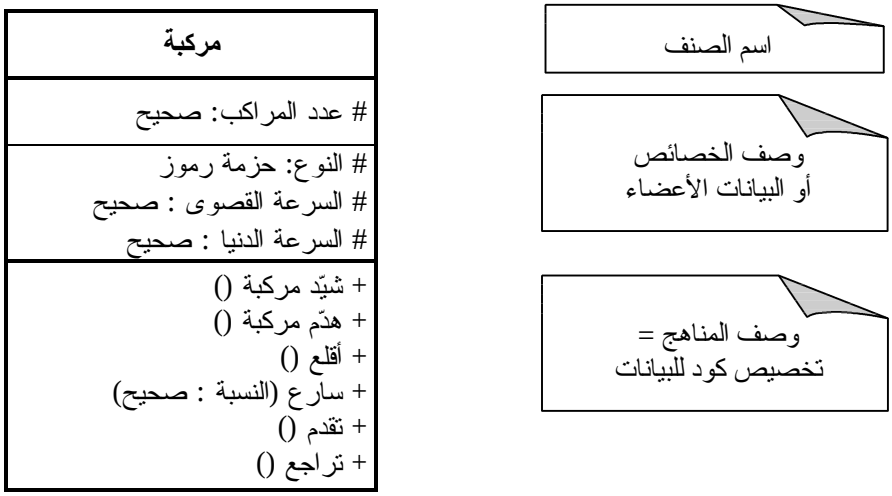
١ مفهوم الكائن:

الكائن **Object** هو أجزاء مترابطة تضم بيانات وكود يعمل في هذه البيانات. أما الصنف **Class** فيمكن اعتباره كقالب حيث من خلاله يمكن خلق كائنات، على كل، نعتبر دائما بأن الأصناف هي أوصاف للكائنات، وهذه الأخيرة هي مثيلات **Instances** عن أصنافها.

لماذا هذا التعقيد؟ يمكن القول أن الصنف يصف الهيكل الداخلي للكائن، البيانات التي يحويها، الأنشطة التي يقدر تحقيقها على هذه البيانات. الكائن هو حالة لصنفه.

لنعتبر مثلا نموذج **مركبة**، كما يمثلها هذا التصميم.





في هذا المثال، **مركبة** قدّمت على شكل حزمة رموز (النوع)، وقيمتين صحيحتين: السرعة القصوى، والسرعة الدنيا. كل هذه البيانات تخص مركبة أيا كانت، بمعنى آخر، كل

كائن من نوع **مركبة** سيكون له نسخته الخاصة من هذه البيانات: نتكلم إذن عن خصائص المثيل (instance attributes).

عملية خلق المثيل (instanciation)، التي تسمح بخلق كائن انطلاقا من صنف تنص على منح قيم مميزة لكل خصائص المثيل.

التصميم السابق يسمح لنا بتقديم الـــUML (Unified Modeling Language)، وهي لغة تسمح بتقديم أنظمة الكائن العالمية تقريبا، والمتداولة في هذه الأيام.

المدخل الحالي لا يسمح إلا بالإطلاع على جزء بسيط من هذه اللغة من خلال تصميمها الثابت والمتعلق بتقديم مختلف الأصناف المتداخلة في النموذج، بمرافقة علاقاتها الأساسية. نلاحظ إذن أن الصنف يتم تمثيله أو تقديمه بمستطيل يتألف من ثلاث أقسام:

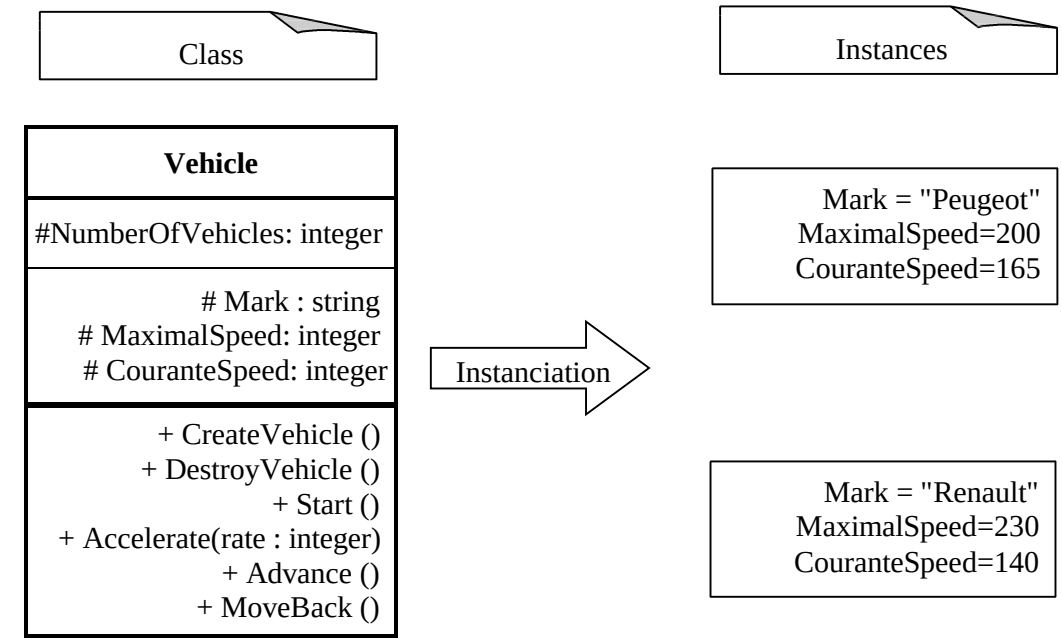
- القسم العلوي يشير إلى إسم الصنف
- القسم الأوسط يحدد الخصائص وأنواعها على شكل:
  IdentifierAttribut : IdentifierType
- القسم السفلي يقدم المناهج مرفقة ببارامتراتها (حججها)، وكذا أنواع القيم المعادة.

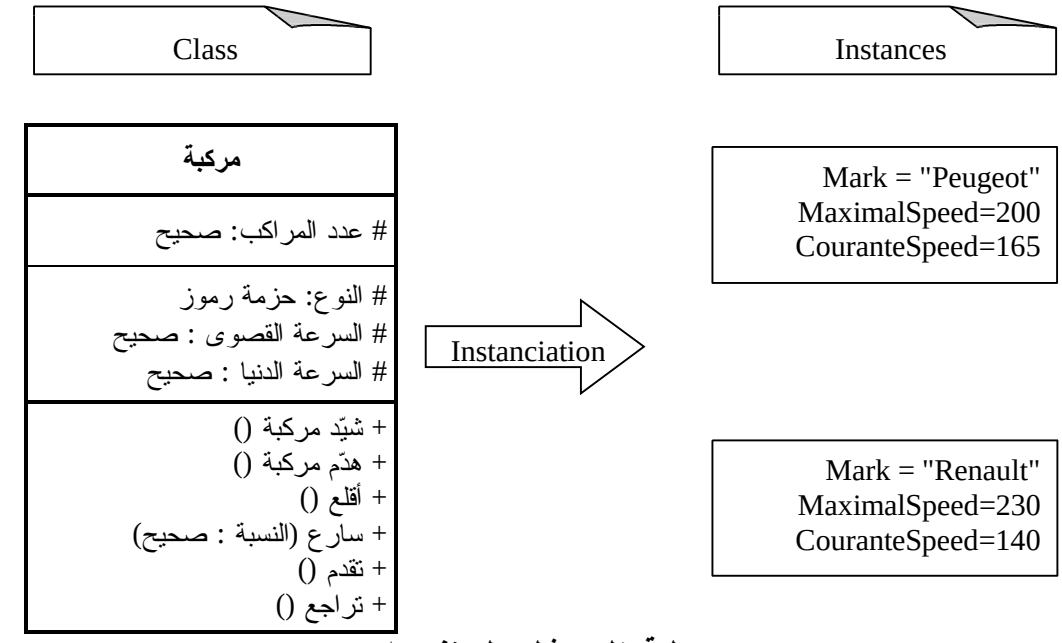
التسطير يشير إلى أن العنصر هو عضو للصنف، سواء تعلق الأمر بالخصائص أو المناهج.

في الأخير، المستطيلات المعلّمة بركن مطوي تخصص للتعليقات أو الملاحظات.

بالمقابل، نلاحظ أن الخاصية عدد المراكب (NumberOfVehicles) المسؤولة عن تحديد عدد المركبات في أي وقت في الصنف. إن هذه الخاصية يتم زيادة محتواها بفضل العملية **شيّد مركبة** (CreateVehicle)، ويتم إنقاص محتواها من خلال العملية **هدّم مركبة** (DestroyVehicle). هذا مثال نموذجي لتقاسم الخواص لمجموعة كائنات تنتمي لنفس الصنف. لهذا يعتبر غير مجدي امتلاك كل كائن لنسخة منفردة من هذه الخاصية، بل ويعتبر عملا خطيرا (تخيل عمليات الإعداد)، إذن من المستحسن أن تتقاسم الكائنات نسخة وحيدة تنتمي للصنف. نتكلم إذن عن خاصية الصنف (Class Attribut).

المثال الآتي يوضح عملية خلق مثيلين لكائنين مختلفين من نفس الصنف:





عملية خلق مثيلين لصنف واحد

نفس المعايير تنطبق مباشرة على المناهج. كذلك، بما أننا ميزنا بين خصائص الصنف، وخصائص المثيل، فسنميز أيضا بين مناهج الصنف ومناهج المثيل.

لنأخذ مثلا المنهاج **أقلع ()Start**، يظهر بوضوح أن هذا المنهاج يطبق على كل عربة بصفة انفرادية. بالمقابل، هذا المنهاج سيستعمل خصائص المثيل (الكائن الجديد)، والتي يجري عليها تطبيقاته، إذن نحن نتكلم عن منهاج المثيل (منهاج يجب لكل مثيل أن يتوفر على نسخة منه).

ولنعتبر مثلا المنهاج **شيد عربة ()CreateVehicle**، هدفه هو خلق عربة جديدة، والذي يمكنه في وقت ثان، من تحديد قيم بدائية لكل خصائص المثيل. إذا اعتبرنا بالتدقيق عملية خلق كائن، نلاحظ أن المرحلة الأولى تتعلق بحجز مكان ذاكري للكائن الجديد. ولكن هذه المرحلة لا علاقة لها بالكائن في حد ذاته، فقط الصنف هو الذي يملك المعلومات اللازمة لعملية الحجز: إذن خلق كائن هو منهاج من مناهج الصنف. نشير كذلك إلى أنه في هذه المرحلة، يستقبل الكائن إشارات إضافية، مثلا: معلومات تشير إلى أي صنف ينتمي هذا الكائن. بالمقابل، فيم يخص عملية تهيئة الخصائص، تطبق فقط في كائن محدد جيدا: والذي هو قيد التشييد. تهيئة الخصائص إذن هي منهاج من مناهج المثيل وليس الصنف.

نستخلص في النهاية العلاقة الآتية: عملية خلق كائن تمر على مرحلتين:

مرحلة أولى، وهي حجز مكان ذاكري من قبل الصنف للكائن الجديد، وتوفير أدنى المعلومات لجعل الكائن ينفذ.

مرحلة ثانية تخص الكائن، وتتعلق بتهيئة خصائص المثيل.

٢ *مفهوم الكبسلة: encapsulation*

بدون معرفة، لقد خطوت خطوتك الأولى نحو أحد عناصر الكبسلة الثلاثة. الكبسلة ترتكز على ثلاثة مفاهيم:

- الكائن يجمع في مضمونه بياناته (خصائصه)، والكود القادر على معالجتها (المناهج).
- تجريد البيانات: هيكل كائن ما لا يظهر من الخارج، شكله يتركب من رسائل غير اصطلاحية، واستقبال لأي رسالة ينجم عنه تنفيذ مجموعة من المناهج.
- تجريد الإجراءات: إذا ما نظرنا من الخارج (بالنسبة لمستخدم الكائن)، المستخدم لا يملك أدنى معلومة حول النشاط الداخلي المطبق، فمثلا: هو لا يدري ما إذا كان العمل المطلوب، يحتاج إلى تنفيذ بعض أو كل المناهج أم لا، أو سيقوم بخلق كائن مؤقت...الخ.

حسب القيم القانونية النموذجية للكائن، خدمات الكائن ليست إصطلاحية (لا يمكن تحقيقها) إلا من خلال رسائل، والتي هي مركبة من:

- إسم
- قائمة بارامترات الدخول
- قائمة بارامترات الخروج

قائمة الرسائل التي يقدر بفضلها الكائن من الاستجابة تمثل واجهته **interface**، إنها القسم العام **public** من الكائن. أما كل ما يخص المعالجة، فيجب أن يبقى مخفيا عن المستخدم النهائي: وهذا ما يمثل القسم الخاص **private** من الكائن، تماما، كل الكائنات المنتمية لنفس الصنف تتمتع بذات الواجهة، بالمقابل كائنين ينتميان لصنفين مختلفين، يقدران على تقديم نفس الواجهة. من وجهات نظر مختلفة، الواجهة يمكن تمثيلها على شكل خاصية متميزة للصنف.

تطبيقيا، في معظم لغات البرمجة الحديثة ذات التوجه الكائني **oop**، الواجهة تمثل قائمة المناهج التي يمكن النفاذ إليها من قبل المستخدم.

من المستحسن إخفاء تفاصيل معالجة الكائنات عن أعين المستخدم، بحيث يسمح ذلك بتغيير مثلا الهيكل الداخلي لبيانات الصنف (تبديل جدول بحزمة رموز) من دون اللجوء إلى تعديل كود المستخدم. أو كمثال آخر: لنعتبر أن صنفا ما يشكل لنا نقطة بإحداثيتين (بعدين)، والتي تعطينا منهاجين يحددان على التوالي موقع النقطة من محوري الفواصل والتراتيب (السطر والعمود)، وبالتالي ليس من الداعي إعلام المستخدم ما إذا كانت النقطة ممثلة داخل الصنف على شكل قطبي أو خطي.

كل اللغات الكائنية التوجه لا تشترط الالتزام بمبدأ الكبسلة (الإخفاء)، إذن هو من خصوصيات وطريقة تفكير المصمم.

٣ *الوراثة: Heritage*

الوراثة هو ثاني مبدأ من المبادئ الأساسية الثلاثة. ويتعلق بترجمة المفهوم الطبيعي للتعميم / التخصيص.

إذن، أغلب الأنظمة الحقيقية تسعى إلى جعل العناصر التي تركبها على شكل تسلسلي (هيارشي). الفكرة الأولى حول هذا الموضوع كانت على علاقة بعلم الأحياء، وبالتفصيل كانت بخصوص تقنية ترتيب الحشرات اعتمادا على معايير مختلفة.

عودة إلى موضوع الوراثة. يرتكز هذا المفهوم على أنه بإمكان كائن ما من الاستفادة من خواص الكائن الأعلى منه، والذي يمكن له أن يضيف مجموعة من الخواص تتعلق به لوحده.

من الجانب الكائني، يمكن ترجمة هذا المفهوم بالطريقة الآتية:

- نخصص صنف إلى المستوى الأكثر عمومية، ونسمي هذا الصنف بالصنف القاعدي، أو الصنف الأب أو كذلك بالصنف الممتاز.
- لكل مفهوم متخصص، نشتق مفهوما قاعديا. الصنف الجديد يطلق عليه الصنف المشتق أو الصنف الإبن أو كذلك بالصنف الفرعي.

الوراثة تعبر عن علاقة تعميم / تخصيص، وبالتالي يمكننا ترجمة كل علاقة وراثية بالجملة الآتية:

الصنف المشتق هو إصدار خاص من صنفه القاعدي

سنقدم مثالين كلاسيكيين لتمثيل فكرة الوراثة:

٣ . ١ . المثال الأول: الكائنات البيانية Graphic objects

لنعتبر مجموعة كائنات بيانية، كل كائن بياني يمكن التعبير عنه بنقطة، وهذه الأخيرة يتم تمثيلها بالإحداثيتين الخطيتين X و Y، ونحدد لها لونها. ماعدا خلق وهدم الكائن، سوف نعين المناهج الآتية لهذا الكائن البياني:

- أنفذ بالكتابة والقراءة إلى الخصائص (البيانات)
- أنشر
- أمحو
- حرك الكائن.

إذن سنحصل على شكل الصنف **ObjetGraphic** المقدم في الصفحة الآتية.

نضيف بعد ذلك صنفين متخصصين: السطر والدائرة **Line** & **Circle**، كل منهما يضيف لذاته بعض الخواص المتعلقة به: القطر **Rayon** بالنسبة للدائرة، الطول والزاوية بالنسبة للسطر **Length** & **Angle**، أيضا يتمتع كل من الصنفين: **السطر والدائرة** بخصائصه (بياناته) الخاصة به والتي تترجم مميزات كل صنف زيادة على خصائص الصنف القاعدي الذي ورثا عنه.

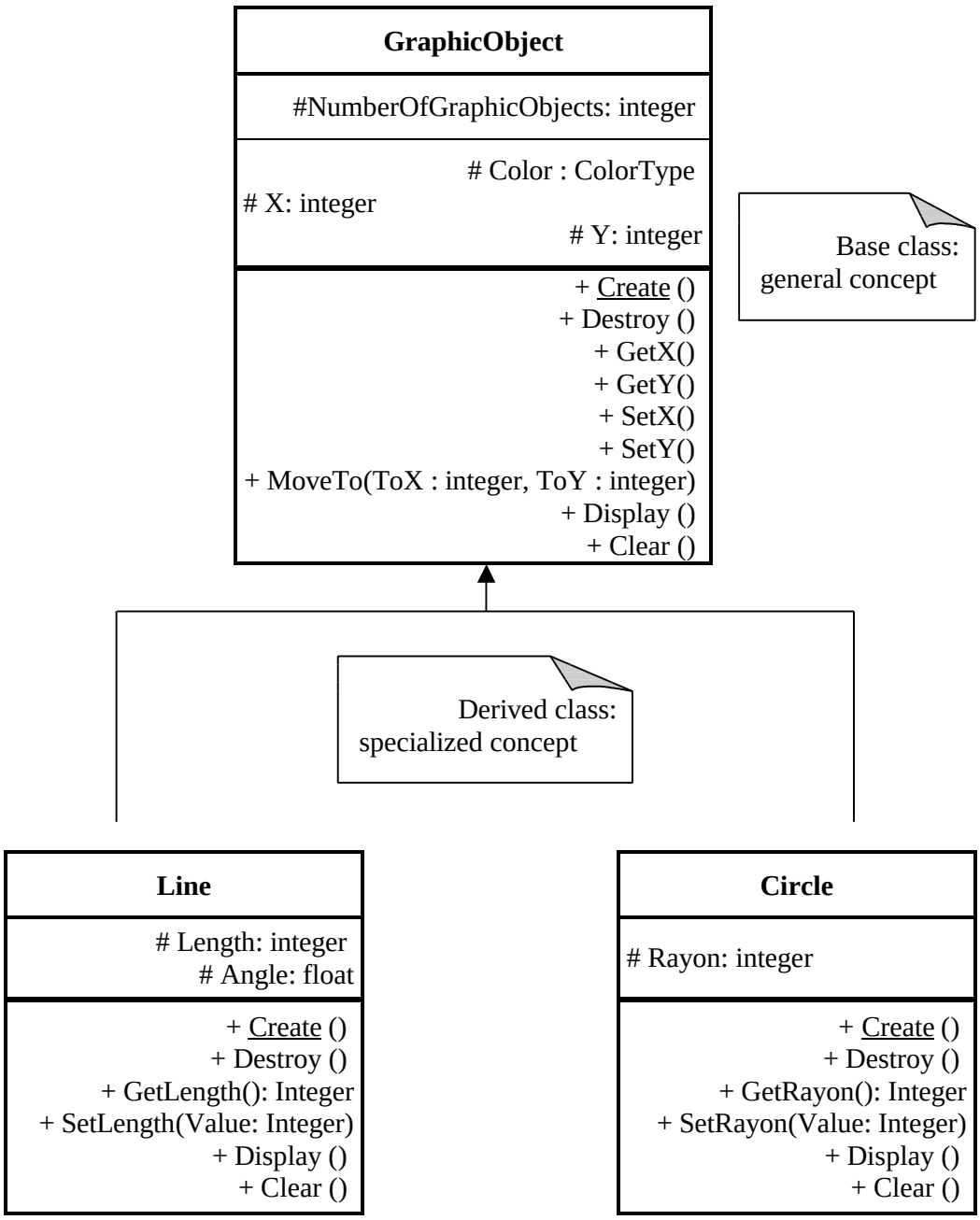
الصنفين: **سطر ودائرة** ليسا بإمكانهما تحقيق الكود للمنهاج **GetX** المسؤول عن تحديد موقع النقطة من محور الفواصل، ولكن بإمكانهما إضافة مناهج أخرى للوصول إلى خصائصهما الجديدة.

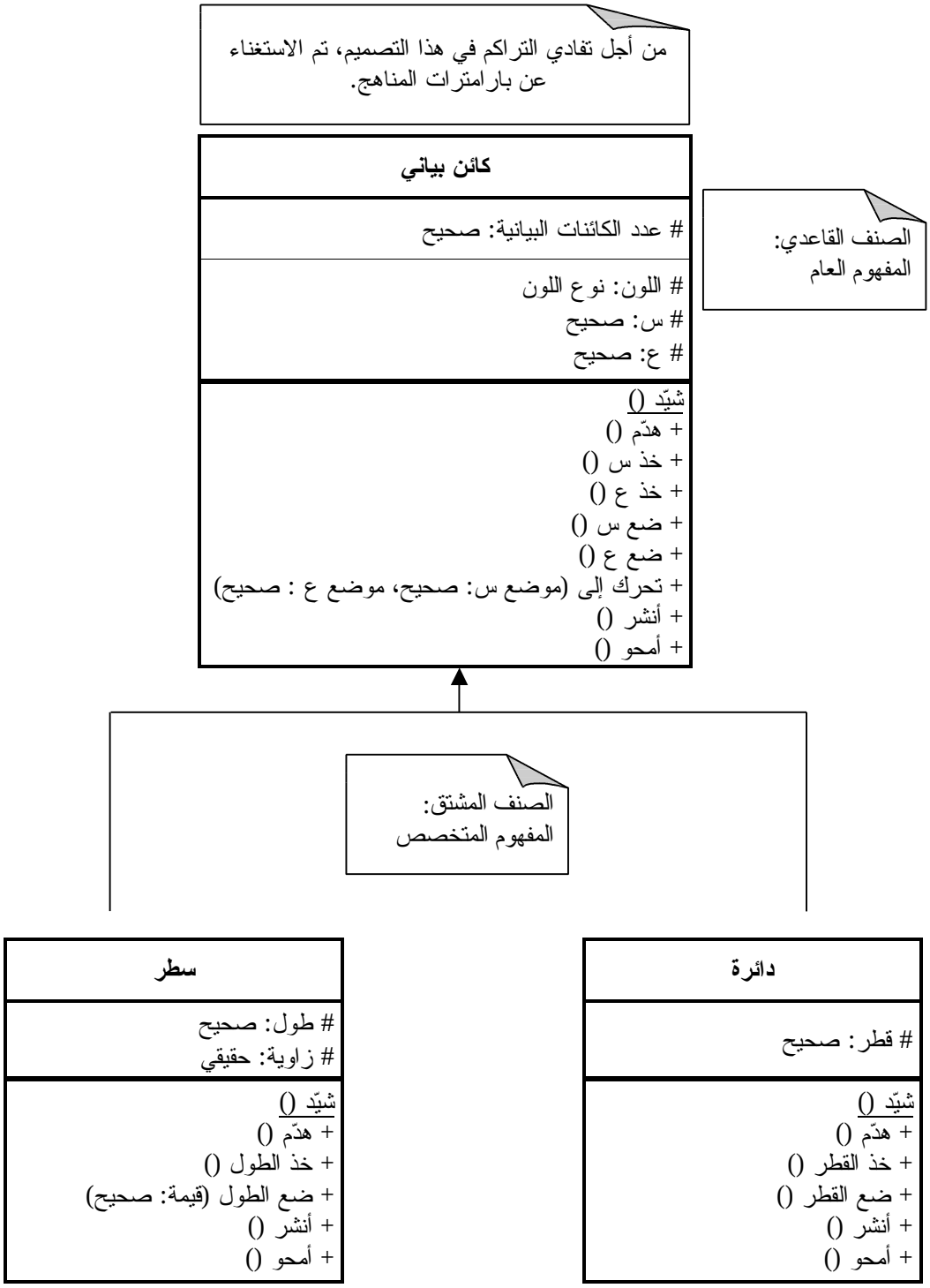
بالإظافة إلى ذلك، إذا ما قمنا بالتمعن جيدا في التصميم، فإننا سنلاحظ أن كلا الصنفين سطر ودائرة قد قام بإعادة تعريف المنهاج **Display** وكذا المنهاج **Clear**، بحيث أن طريقة نشر السطر مثلا تختلف عن طريقة نشر دائرة، ونفس الفكرة تنطبق على طريقة المحو: إنه تعدد الصفات **Polymorphism** المطبق على المنهاجين **Display** و **Clear** في إطار الصنفين **سطر ودائرة**.

سنعود بالتفصيل حول مفهوم تعدد الصفات، ولكن في الوقت الحالي يكفيك معرفة أن المنهاج (أو الإجراء / الدالة) يمكنه اتخاذ عدة أشكال:

- شكل قوي: التحميل الزائد **overload** الذي يسمح باستخدام نفس الإسم للمنهاج / الإجراء / الدالة مع قائمة من البارامترات المختلفة.
- الشكل القوي لتعدد الصفات الذي يتعلق بإعادة تعريف المنهاج التابع للصنف الأب من قبل الأصناف المشتقة منه مع استعمال نفس التوقيع (نفس قائمة البارامترات ونفس نوع قيمة العودة).

من المهم ملاحظة أن توقيعات (قائمة البارامترات ونوع قيمة العودة) المنهاجين **أنشر** و**أمحو**، هي نفسها في الصنف الأب أكثر منه في الأصناف المشتقة. هذا يسمح بطلب المنهاج في أي كائن من نفس التسلسل وبنفس الأسلوب، ومن دون الحاجة لمعرفة إلى أي صنف ينتمي المنهاج المطلوب. قوة تعدد الصفات polymorphism غير محدودة.





تسلسل الصنف GraphicObject

حسب الـــ UML، علاقة الوراثة يتم الإشارة إليها بسهم ذو نهاية مثلثية، وحيث الاتجاه يكون نحو الصنف الأب.

المناهج المكتوبة بخط مائل هي مجردة. فنستنتج مباشرة أن الأصناف ذات الإسم

المائل (italic) هي أيضا مجردة، وبالتالي لا يمكننا خلق مثيلات عنها مباشرة.

للتذكير، المناهج والخصائص المسطرة هي أعضاء للصنف.

أجلب انتباهك إلى أن المنهاج **MoveTo** لم يعرّف مرة ثانية، إذن، بأسلوب أدبي، يمكننا تركيبه على هذا الشكل:

```
method GraphicObject :: MoveTo (Pos X: Integer, Pos Y: Integer)
{
        [object Clear]
        [object Set X : Pos X]
        [object Set Y : Pos Y]
        [object Display]
}
```

الكود العام للمنهاج MoveTo

إنه من السهل ملاحظة أن هذا التركيب سليم من حيث أن منهاج محو السطر **Clear** يتم طلبه إذا ما تم تنفيذ المنهاج **MoveTo** لأي كائن مشتق من الصنف **Line**، ونفس الشيء للكائنات المشتقة من الصنف **Circle**. مرة أخرى نستعمل هنا مبدأ تعدد الصفات. فإذا تم تطبيق المنهاج **MoveTo** على كائن من نوع **Circle**، فإن هذا المنهاج يقوم بطلب **Clear** و **Display** التابعين للصنف **Circle** (نشير إلى Circle::Display و Circle::Clear)، أما لو تم تطبيق المنهاج **MoveTo** على كائن من نوع **Line**، ففي هذه الحالة يقوم هذا المنهاج بطلب **Clear** و **Display** التابعين للصنف **Line** (وهما المنهاجين Line::Clear و Line::Display).

مع بساطة هذا المثال، نستخلص جملة من المزايا والتي نورد بعضها:

- الكود يكون أقل حجما لأننا جمعنا الأكواد المتشابهة وجعلناها في الأصناف القاعدية والأكثر عمومية.
- على مستوى الأصناف المشتقة، فقط نكتب الكود الخاص للصنف، بحيث ليس من الداعي إعادة نفس التعليمات عند كل مرحلة، وبالتالي يسير التطوير بوثيرة سريعة.
- قولبة مفهوم ما، يجعل نظام العمل مهيكلا بأسلوب جميل، وبالتالي يمكن تطويره.
- ميكانيزم تعدد الصفات القوي يعتمد تماما على الوراثة، كما سنراه لاحقا.
- كود الأصناف العليا في التسلسل (الأصناف العامة) يتم استخدامه غالبا، مما يسهل عملية اكتشاف الثغرات والأخطاء **debug**.
- إذا كان التسلسل مصوّرا بطريقة جيدة، فإنه يصير من السهل إضافة أصناف جديدة، مع الأخذ بعين الاعتبار الاختلافات الموجودة بين الصنف الجديد والأصناف الموجودة ضمن التسلسل: نتكلم إذن عن البرمجة المتنوعة differential programming.

٣ . ٢ . *المثال الثاني: قولبة حظيرة للعربات*

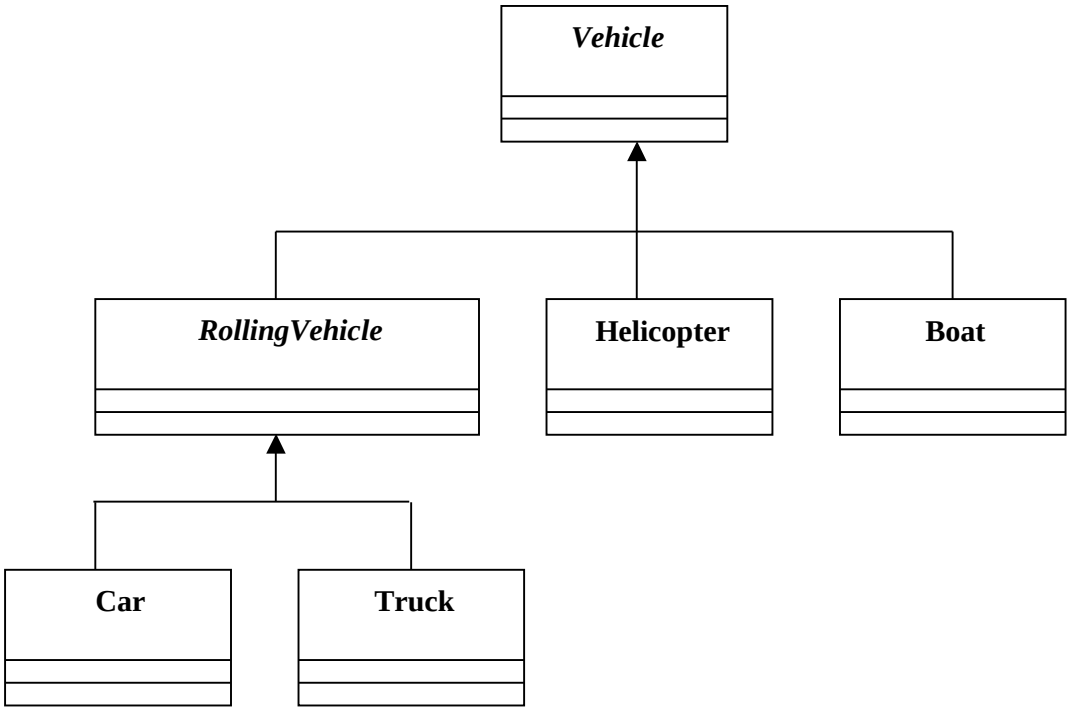
سيبين لنا هذا المثال أسلوب استخدام التعميم/التخصيص من أجل تركيب نظام ذو توجه كائني ناجع.

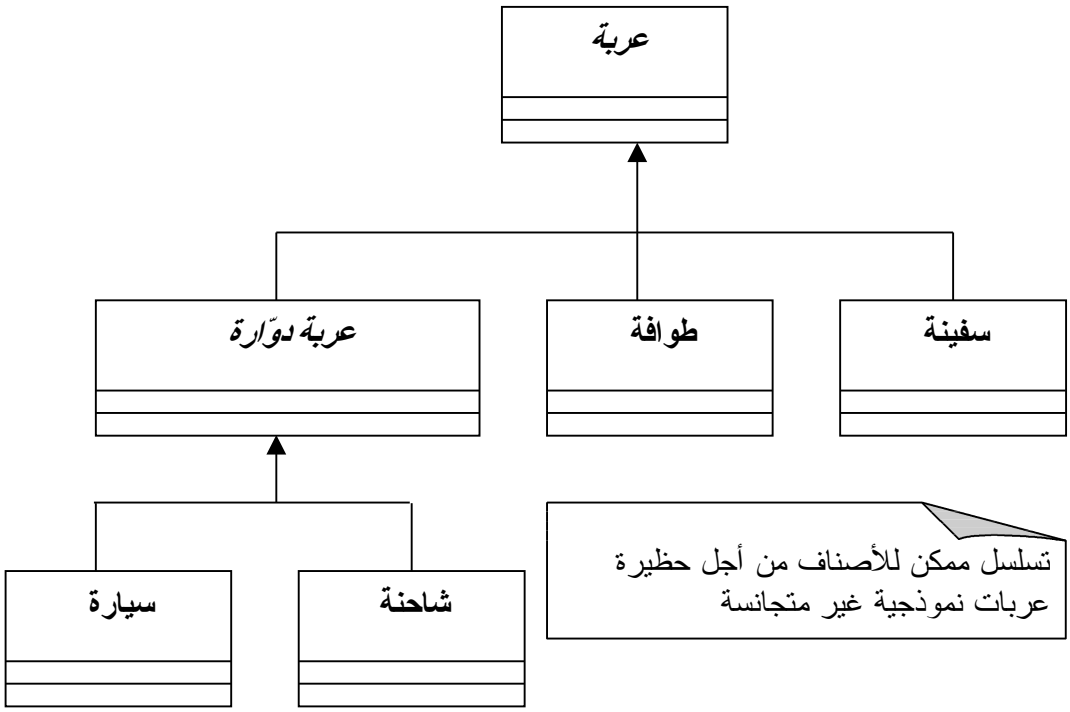
تتوفر مؤسسة على حظيرة عربات تضم:

- سيارات Cars
- شاحنات Trucks
- طوافات Helicopters
- سفن Boats

تود هذه المؤسسة إنجاز نموذج يمكن من خلاله وضع أي نوع من العربات ضمن القالب الملائم له. الهدف هو خلق نظام أصناف يتم عن طريقه التعامل بنظام مع خصوصيات كل نوع من العربات. ننطلق من الأنواع التي يمكن قولبتها، فنلاحظ أن السيارة والشاحنة يمكن اشتقاقهما من نفس الصنف، ونترك الصنفين الآخرين: طوافة وسفينة في الجانب الآخر. وزيادة على ذلك، فعلى الرغم من الاختلاف الواضح بين العربات، إلا أنها تتقاسم العديد من الخصوصيات من حيث شكلها المتحرك. وأيضا ببعض النشاطات كالإقلاع، الإسراع، التباطؤ أو التوقف. فكل هذه الأعمال لها معان لكل نوع من أنواع العربات. أيضا، كل أصنافنا لها أصل أو جد واحد، والذي نسميه: **عربة**.

إذن نحصل على النموذج الآتي:

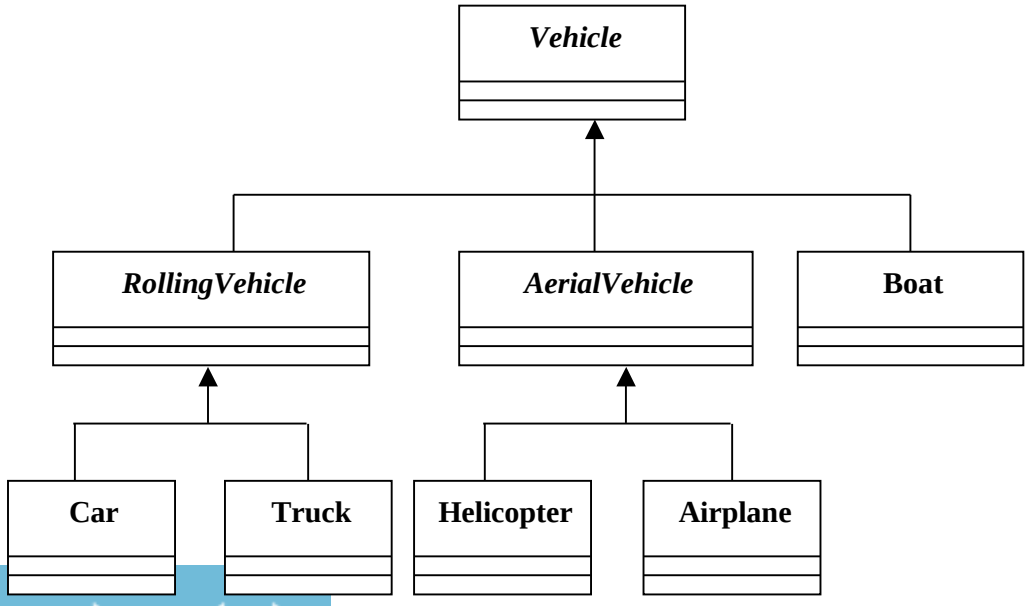


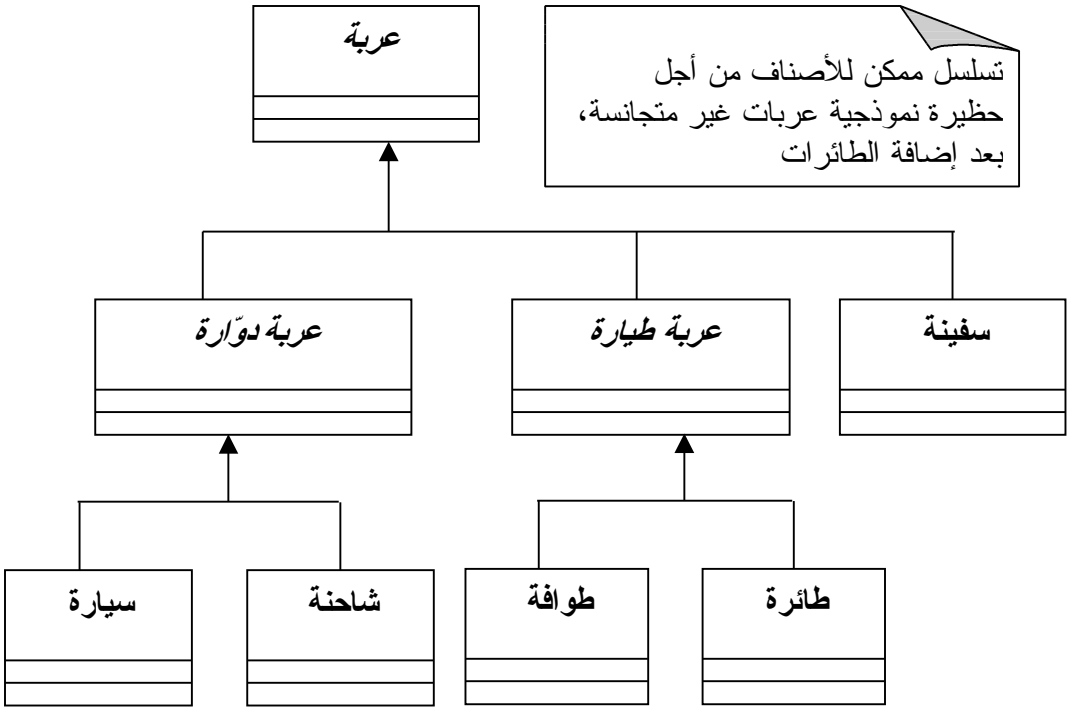


نموذج لحظيرة عربات

من أجل تحقيق هذا النموذج، اتبعنا أسلوب التعميم: من خلال مجموع الكائنات، استخلصنا العناصر المشتركة التي تسمح لنا بجمعها ووضعها في إطار عام. هذا الأسلوب يسمى التعميم، وهو نمط مستحسن لخلق تسلسل أصناف.

أسلوب التخصيص يستخدم بكثرة إذا تعلق الأمر بإدراج أصناف جديدة ضمن نظام موجود مسبقا. على كل، إذا سعت الإدارة إلى اقتناء بعض الطائرات، فإنه يكون بالإمكان إضافة صنف جديد (من خلال التعميم) **AerialVehicle**، والتي نشتق منها **طائرة** و **طوافة**، فنحصل إذن على التصميم الآتي:





هذا النوع من الترتيب يأخذ أحيانا إسم taxonomy، فمن أجل وضع تقديم لهذه المبادرة: يبحث المتخصصون في الحشرات دوما عن وسائل تطبيقية مترابطة لجعل تصنيف الحشرات يسير في أحسن ما يمكن.

مبدأ التعميم/ التخصيص هو بديهي وقوي لأنه يسمح بتعريف التصرفات المتشابهة على طول شجرة الإشتقاقات، كل صنف فرعي يمكننا الاختيار بين إعادة تعريفه أو توريثه من الصنف الأعلى منه.

٣ . ٣ . الأصناف المجردة: abstract

بقراءة متمعنة، لابد وأننا لاحظنا أن الأصناف ***عربة***، ***عربة دوارة*** و ***عربة طيارة***، مكتوبة بخط مائل في التصميم السابق. هذا ليس من باب تنميق التصميم، ولكن لأن هذه الأصناف هي مجردة.

نقول عن أصناف أنها مجردة إذا كانت لا تمنح عملا برمجيا implementation لمناهجها، ومناهجها تصير أيضا مجردة. ربما مازال الغموض؟ على كل، الصنف المجرد، لا يمكنه أن يخلق مثيلات (كائنات)، يمكن ذلك للأصناف المشتقة منه من خلال توفير الكود لكل من المناهج التي كانت مجردة، وبالتالي يزول عنها التجريد وتصير قادرة على القيام بعمل برمجي، نتكلم إذن عم أصناف حسية concrete. الأصناف الحسية هي الوحيدة القادرة على خلق مثيلات.

ما هو هدف الأصناف المجردة؟ هو خلق إطار عمل للأصناف المشتقة من خلال تقديم مجموعة من المناهج التي نجدها على طول التفرع (التسلسل في الأصناف

الجد/الأب/الإبن...)، تعتبر هذه الآلية أساسية لتحقيق تعدد الصفات polymorphism.
بالمقابل، إذا ما اعتبرنا الصنف **عربة**، فمن الطبيعي أنه لا يمكننا أن نخلق مثيلات عنه: **عربة** لا ترمز إلى أي كائن حسي، ولكن ترمز إلى مفهوم كائن يقلع ويبطئ ويسارع أو يتوقف، سواء كان هذا الكائن سيارة أو طائرة أو شاحنة.

تحديد ما إذا الكائن حسي أو مجرد، هذا يتعلق غالبا بدفتر حمولة البرنامج. بدراسة متعمقة لهذا الأخير، يمكن تحديد ما هي الأصناف الحسية: التي تمتلك مثيلات. ومن خلال هذه الأخيرة يمكن تحديد الأصناف المجردة والتي تسمح بالاستفادة من تعدد الصفات polymorphism، وهذا باتباع أسلوب التعميم.

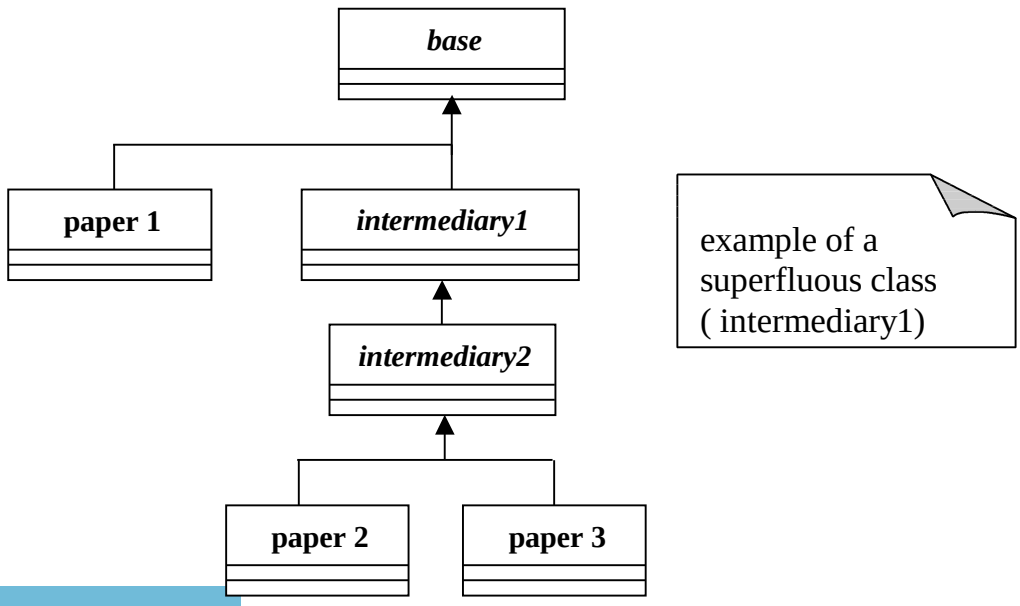
٣ . ٤ . *الصعوبات المرتبطة باستخدام الوراثة:*

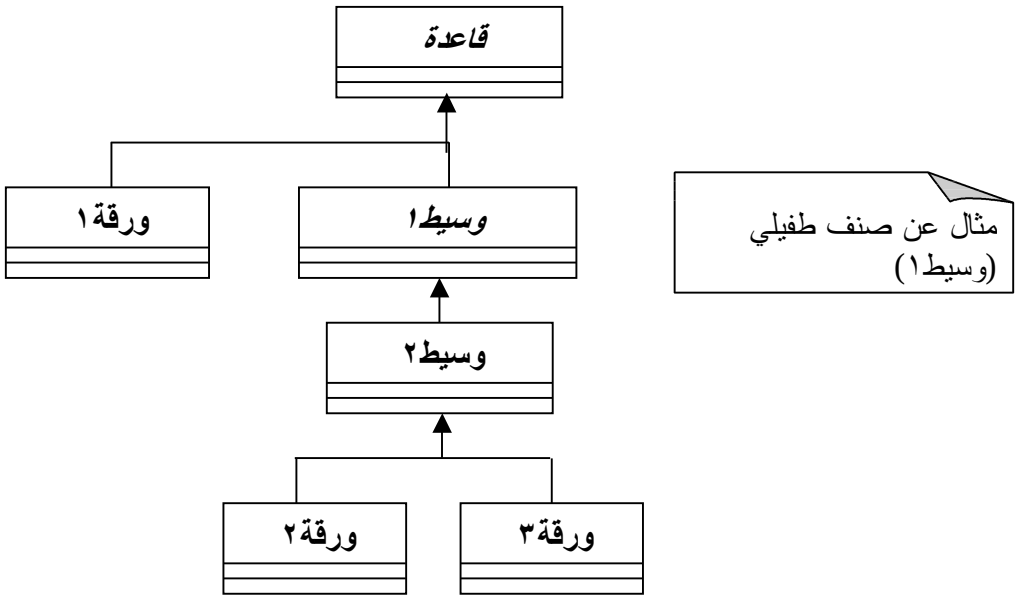
سنرى نموذجين حيث كان التقسيم وهمي. غالبا ما يكون تحديد التسلسل المناسب صعبا. القاعدة سهلة، علاقة الوراثة يجب أن تترجم إلى " الصنف المشتق هو إصدار خاص من صنفه القاعدي ".

٣. ٤. ٠١. تسلسل مكثف جدا:

يجب الانتباه إلى عدم تكثيف وإثقال التسلسل بالاشتقاق الغير منظم. فلنعتبر مثلا الصنف **حيوان**، والذي منه نشتق الصنفين **قط** و **كلب**. فلحد الآن، لا يمكننا التعليق، لأننا نفهم بأن الاختلاف الموجود بين هذين النوعين يتطلب منا خلق صنفين مختلفين. بالمقابل، يعتبر اشتقاق الصنفين **كلب_أسود** و **كلب_أصفر** من الصنف **كلب**، سوء للتقدير، فبسبب لون الشعر، تم خلق صنفين، أو عدة أصناف، مما سيثقل التسلسل لأسباب بسيطة. كان من الأحسن دمج صفة اللون (أو بعض الصفات البسيطة) ضمن خصائص (بيانات) الصنف الأساسي **كلب**.

بنفس الأسلوب، يجب الحرس على عدم دمج أصناف وسيطة كثيرة، فالتصميم الآتي يبين أن هناك صنف وسيط غير ضروري ولا يملك أي اشتقاق حقيقي، لذا يجب حذفه.





إذن، إذا كان الصنف **قاعدي** يمتلك صنفين ابنين، الصنف **وسيط١** لا يعمل إلا على إثقال غير ضروري للتسلسل. إذن هو صنف مجرد (كما نشير إلى خطه بنوع مائل) ولا يمكنه أن يمتلك مثيلات (كائنات)، وفي نفس الوقت لا يُشتق إلا مرة واحدة. يمكننا إذن حذفه مع إضافة خصائصه في الصنف **وسيط٢**. هذا المثال يسمح بزيادة قاعدة لغوية: *نسمي ورقة الصنف الذي لا يملك مشتقات*.

٣. ٤. ٠٢. وراثة التشييد: construction inheritance

وراثة التشييد هو مثال آخر سيئ فيم يخص استعمال الوراثة، حيث ينص على اشتقاق صنف، مع إضافة له خصائص تغير جذريا من المفهوم المستخدم. فكمثال، إذا ما تم اعتبار أن المستطيل هو عبارة عن سطر، ثم نقوم بإضافة بعد ثاني له. على كل، يمكن ملاحظة أن الوراثة في الحالة الأخيرة صارت غير مناسبة، لأن الجملة: " المستطيل هو إصدار خاص من السطر " ليس لها أي معنى.

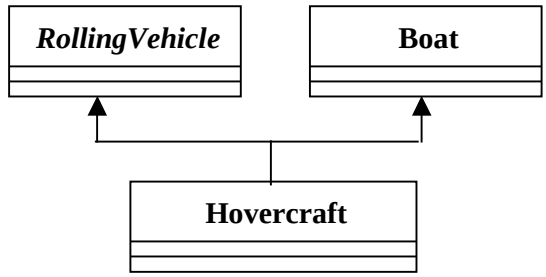
٣. ٤. ٠٣. التصورات الغير متناسقة: the conceptual incoherence

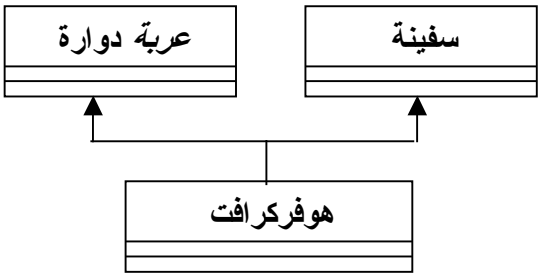
تؤدي الوراثة بعض الأحيان إلى تصورات ضالة. فلنعتبر مثلا الصنف **طائر** المزود بالمنهاج **يطير**، نستطيع مثلا اشتقاق الصنف **دجاجة** التي هي من فصيلة الطيور، سواء بأسلوب الوراثة أو عن طريق إعادة التعريف، ولكن كل منا يعلم أن الدجاجة لا تطير رغم أنها من فصيلة الطيور. إذن في هذه الحالة نستطيع أن نقول "دجاجة هي طائر متخصص". كذلك، تطرح بعض اللغات الوراثة الاختيارية: المبرمج مدعو لاختيار ما هي المناهج والخصائص التي يمكنها أن تورث. وإن لزم الأمر محو منهاج الطيران من تسلسل **طائر**، فإنه لا يطرح أي مشكلة. بالمقابل، هذا يعتم كلية مفهوم تعدد الصفات polymorphism الذي ينص على أن منهاجا موجودا في الصنف القاعدي يمكن أن يكون موجودا في الأصناف الوريثة.

٣ . ٥ . الوراثة المتعددة : the multiple inheritance

الوراثة المتعددة هي توسع للوراثة العادية، حيث نسمح لصنف من أن يمتلك العديد الأصناف الأباء لكي نعطي قالبا للتعميم المتعدد.

إذا ما أشكل علينا الفهم، فلننتبه إلى هذا المثال: لنفرض أننا نريد زيادة الصنف **هوفركرافت Hovercraft** (لم أجد لها مصطلحا مناسبا) لنموذج حظيرة العربات السابقة. ونعلم أن هوفركرافت هو سفينة (باخرة، عبارة ..)، وفي نفس الوقت عربة برّية، إذن نقدر على أن نقولبها على هذا الشكل:

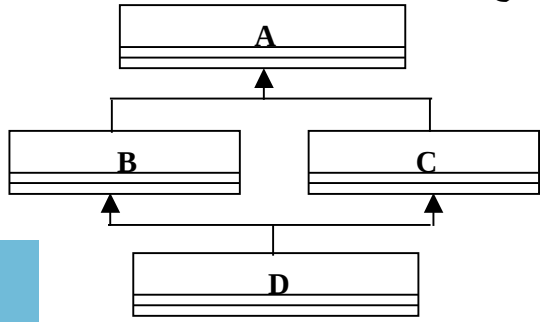




مثال عن استعمال الوراثة المتعددة

مشاكل استعمال الوراثة المتعددة لا تنتهي عند هذا الحد، بل تطرح نفسها في أشكال أخرى، فمثلا لو كان عندنا صنفين قاعديين يملكان خصائص أو مناهج متشابهة، فإننا نجد أنفسنا في مواجهة مشكلة تشابك في التسمية، والتي يجب أن نحلها. بعض اللغات تشترط إذن لحل هذا الإشكال إلحاق المنهاج أو الخاصية باسم الصنف التابعة له.

مشكلة أخرى، وهي أنه أحيانا تتحول الوراثة المتعددة إلى وراثة تكرارية. ففي المثال الآتي نجد الصنف **D** يرث نسختين من الصنف **A**، واحدة من خلال الصنف **B**، والأخرى عن طريق الصنف **C**، والسؤال المطروح بأي من النسختين سيحتفظ؟ بعض اللغات كالسي بلس بلس تقترح آلية تسمح بمعالجة هذه المعضلة الشائكة إذا ما حدثت.



العديد من مكتبات الأصناف تستخدم صنف قاعدي مجرد لأجل الاستفادة من آليات تعدد الصفات polymorphism. وأيضا قد نقع في مشاكل استخدام الوراثة التكرارية كلما استعملنا الوراثة المتعددة. الواجهات الموفرة من قبل بعض اللغات التي تنفي الوراثة المتعددة (كالسي بلس بلس و الجافا) تقترح تناوبا هاما ومناسبا لتعدد الوراثة.

٣ . ٦ . *الواجهات*: the interfaces

المشاكل المرتبطة بالوراثة المتعددة دفعت بعض المصممين للغات إلى اقتراح عدة حلول. إلى أن رأينا ظهور آليات أخرى *كالواجهات*. الواجهة شبيهة بصنف من دون خصائص (بيانات)، ولكنها تقدر على احتواء ثوابت، أما مناهجها فهي كلها مجردة.

زيادة على مميزات الوراثة التي يتمتع بها الصنف، فإنه يقدر على وضع الواجهة قيد التنفيذ Implement إذا كانت تملك تنفيذا لكل المناهج الموجودة فيها. هذه الآلية قوية للغاية لأنها تخلق علاقات قوية بين مختلف الأصناف المنفذة لنفس الواجهات من دون أن تكون لها علاقة أبوية. على الخصوص، المناهج الموجودة ضمن الواجهات هي متعددة الصفات polymorphs، لأنها تنفذ بصفة محايدة في كل صنف يستخدم نفس الواجهة.

بالمقابل، كل صنف يمكنه تنفيذ ما يحلو له من الواجهات. هذه الآلية الصادرة عن **Smalltalk** تم تبنيها من قبل لغة السي الكائنية (أو كما يسميها البعض: الشيئية) والجافا.

من أجل تثبيت الأفكار، لنفرض مثلا أن نظام كائنات يريد قولبة غواصة نووية، وحيث بعض مركباتها تابعة لفروع تسلسلية يجب أن تتواجد على التصميم. فبدل اشتقاق كل الأصناف لنفس العائلة، والتي تقترح عرضها على التصميم (بعض الأصناف لا يجب أن تعرض)، وهذا ما يعتبر تصورا سيئا، فبدل ذلك، يستحسن ربط واجهة قابلة للعرض إلى التي يجب أن تظهر في التصميم. نشير أيضا إلى أنه لو كانت كل الأصناف المشتقة من صنف قاعدي يقترح مناهج للرسم، فإن كل الأصناف الغير بيانية ستثقل بالعديد من المناهج الغير ضرورية.

التمييز بين الاشتقاق واستخدام الواجهات، ليس بالبساطة بما كان، ولكنه يتعلق بنظرة المتصور (المصمم)، وأيضا ببيئة البرنامج المراد إنجازه. إذن، بالرجوع إلى مثال **هوفركرافت**، هل يجب اشتقاقه فقط من ***عربة دوارة***، وتزويده بواجهة من ***عربة بحرية***، أو إجراء وراثة من **سفينة** وتزويده بواجهة من ***عربة برية***، أو أيضا، خلق صنف جديد مستقل، وتزويده بالواجهتين معا. الإجابة عن هذا السؤال ليست بسيطة، ولكن تتعلق بالأولويات التي نريد إعطاءها للنموذج: المعيار الأفضل هو عمل الوراثة. مرة أخرى يطرح مشكل طبيعة التقسيم، والتي تتعلق بشخصية صانع النموذج، وببيئة التطبيق المراد.

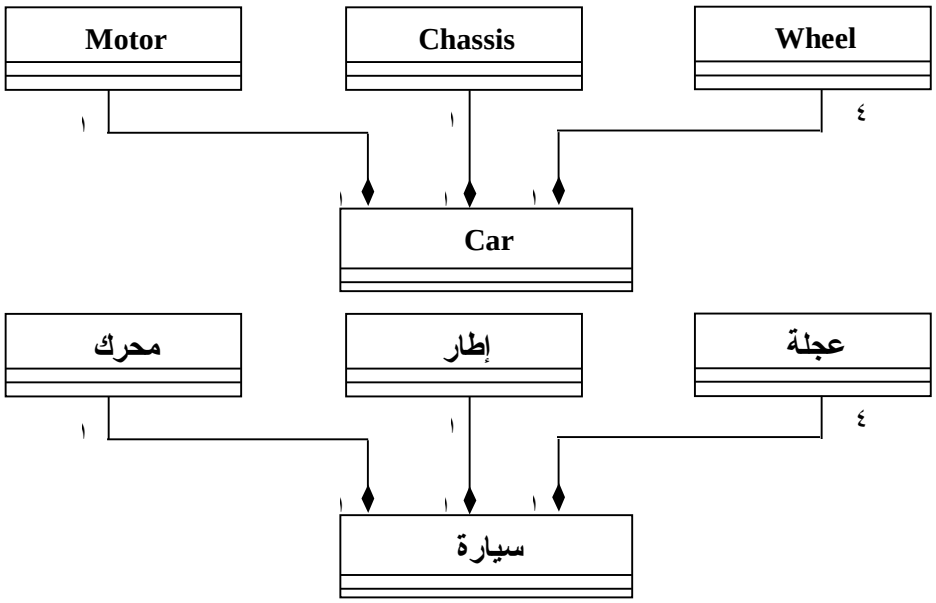
٤ *التركيب* aggregation:

٤ . ١ . *تعريف*:

التركيب هو نوع آخر من العلاقة بين صنفين، والتي يترجم هذه المرة العلاقة "***مكوّن من...***" أو "***يملك...***" أو أيضا "***له...***". فمثلا، في نظام ميكانيكي، يمكننا أن نقول أن الصنف **سيارة** مكون من صنف من **محرك**، أربعة أصناف من **عجلة**، وصنف من **إطار**. خلق مثيلات

يمر بصفة إلزامية إلى استعمال الخصائص والتي هي في حد ذاتها كائنات أو مؤشرات على كائنات أو أيضا مثيل عن صنف حاو يحقق هو كذلك مبدأ التركيب. الشيء الهام في التركيب هو كارديناليته. لنعتبر مثلا تركيب عجلات من سيارة. أي سيارة تتركب من أربع عجلات (مع إهمال عجلة النجدة، أو السيارات الشاذة ذات الثلاث عجلات)، وأي عجلة لا يمكنها أن تُمتلك من أكثر من سيارة. كارديناليةالتركيب هي إذن ١ من جانب المركِّب، و٤ من جانب المركَّب.

المثال الآتي يسمح بتثبيت المفاهيم:



مثال لتركيب بالنسبة للصنف عربة

قواعد أنظمة الكائن العالمية UML تستعمل سهما يكون شكل مقدمته معين لتقديم التركيب aggregation. المعين يكون جهة المركِّب. أما الكارديناليات فتكون محددة.

- كارديناليةالمركِّب تكون بجانب المعين (في البيان: عجلة تنتمي لسيارة واحدة).
- كارديناليةالمركَّب تكون بجانب الخط (في البيان: سيارة تمتلك أربع عجلات).

٤ . ٢ . *التركيب كتناوب للوراثة المتعددة أو الواجهات:*

أحيانا يكون بالإمكان ترجمة بمفهوم التركيب مفاهيم تظهر في الوراثة المتعددة أو في الواجهات. فلنفرض مثلا نظام عسكري يجمع طائرات ورادارات. وفجأة نريد دمج **أواكس AWACS**، والتي هي طائرة تمتلك رادارا. فكيف نقولب (نصنع نموذج أو قالب) هذه الوضعية؟

١ الوراثة المتعددة: نشتق الصنف **أواكس** من الصنفين **رادار** و **طائرة**.

٢ نستعمل الواجهات:

نشتق **أواكس** من **طائرة**، ونظيف لها واجهة **لاقط** (صنف يجمع كل أنواع الرادارات).

نشتق **أواكس** من **رادار**، ونظيف لها واجهة **آلة طيارة** (صنف يجمع كل أنواع الطائرات).

نخلق صنف **أواكس** جديد ونزوده بواجهتين **لاقط** و **آلة طيارة**.

استعمال التركيب:

نشتق **أواكس** من **طائرة**، ونظيف لها خاصية **رادار**.

نشتق **أواكس** من **رادار** لها خاصية **طائرة**

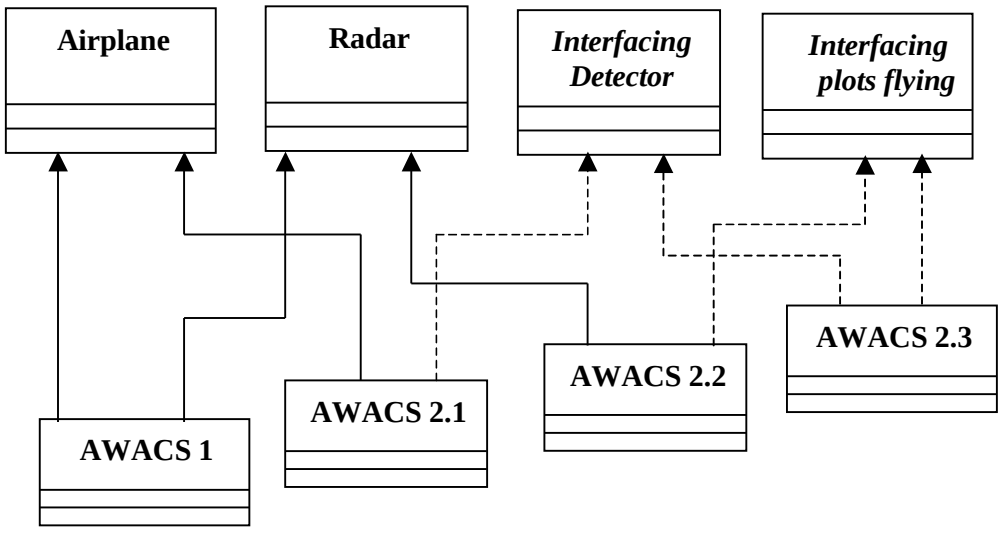
نخلق صنف **أواكس** جديد ونزوده بخاصيتين: **رادار** و **طائرة**.

...

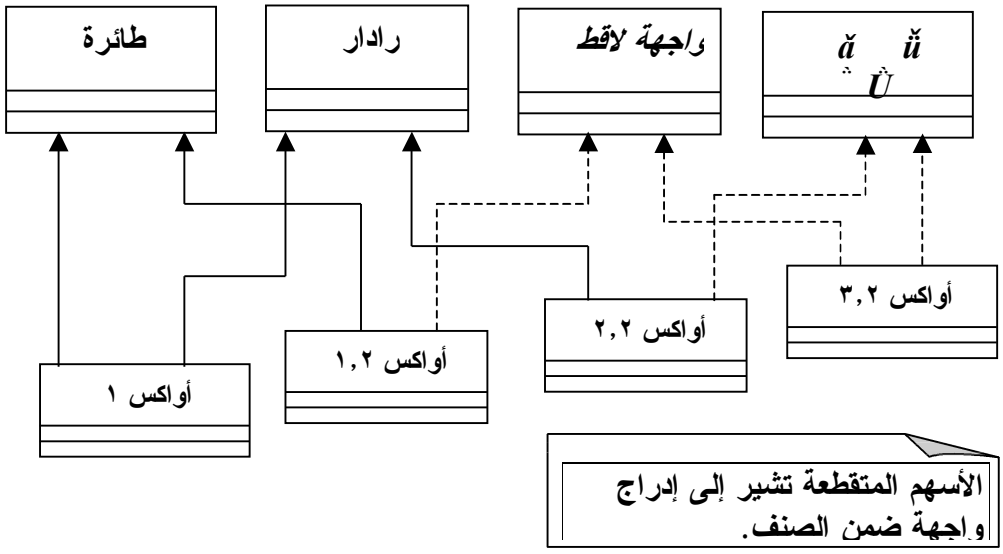
هذه الإمكانيات المتعددة توضح مدى غنى الكائن وفعاليته. وإذا ما ظهر أن بعض النماذج أحسن من غيرها، فإنها ليست نتيجة ثابتة، فبتغير المعطيات تتغير الطرق. كما أنه يمكن استخدام الواجهات مع التركيب في آن واحد، أو الوراثة مع التركيب كما سنشير إلى ذلك.

عدا ذلك، أي مصمم رادارات يفضل الحالات ١، ٢,٢، ٣,٢، والتي ليست بالضرورة نفس فكرة صانع الطائرات الذي قد يفضل حالات أخرى والتي قد تكون: ١، ١,٢، ١,٣.

التصميمات الآتية تبين مختلف الحالات التي يمكننا تحقيقها.

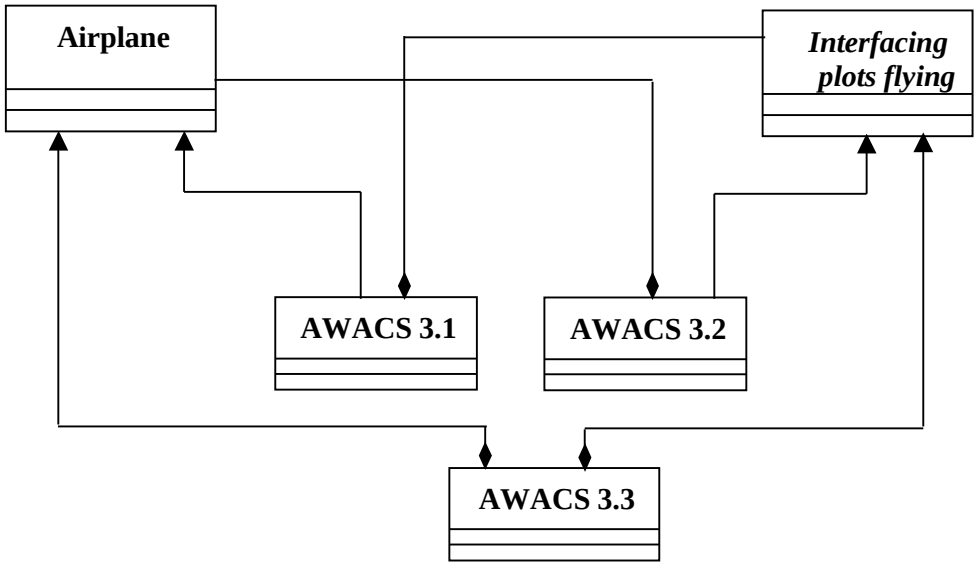


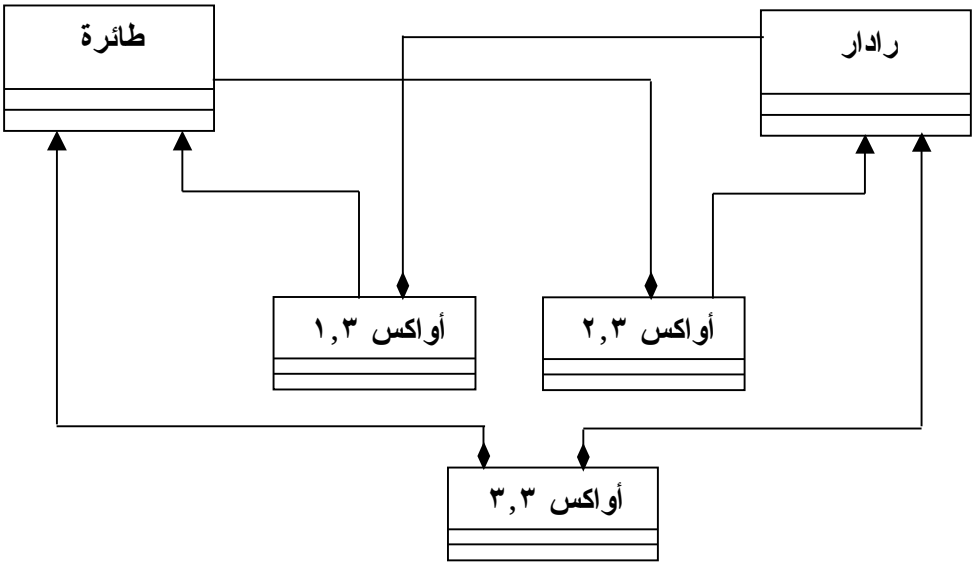
قولبة أواكس باستخدام الواجهات



في الوقت الحالي، لا يوجد مبدأ ثابت فيم يخص تقديم وتمثيل الواجهات في لغة أنظمة الكائن UML. أيضا، الوسيلة الأكثر بساطة تتمثل في التصريح بصنف مجرد (من دون خصائص) وحيث الإسم يكون مسبوقا بالمصطلح Interface.

استخدمنا سهما يشبه ذلك المستعمل في تمثيل التعميم/التخصيص ولكن مع خط متقطع لتوضيح أنه إدراج لواجهة من قبل الصنف. رأس السهم يكون موجه نحو الواجهة.





قولبة أواكس باستخدام التركيب والوراثة

٥ تعدد الصفات Polymorphisme :

٥ . ١ . تعريف:

تعدد الصفات هو المبدأ الثالث من المبادئ التي يرتكز عليها وجود الكائن. أو بالأحرى هو المبدأ الأكثر تأثيرا والأقوى. وكما يشير إليه اسمه، فإن تعدد الصفات يسمح لأي منهاج من أن يتخذ عدة وجوه في أصناف مختلفة. فمن خلال اللغات، تعدد الصفات يمكن التعبير عنه في مجموع أصناف نظام ما، في حين يصنفه آخرون ضمن الأصناف التابعة لنفس التسلسل.

٥ . ٢ . قوة تعدد الصفات:

سوف نبين قوة تعدد الصفات من خلال معالجة لمثال **كائن بياني GraphicObject** . أي صورة يمكن رؤيتها كمزيج من الأشكال الهندسية: مربعات، مثلثات، دوائر وأشياء أخرى يمكن اشتقاقها من الصنف **كائن بياني**. تركيب على هذا الشكل ممكن من خلال مبدأ التوافق التنازلي للمؤشرات (notion of pointers downward compatibility). على كل، مؤشر (أو في بعض اللغات "**المرجع Reference**") على كائن لصنف متخصص يمكن دائما أن يؤشر على كائن من صنف عام.

إذا ما أردنا تخطيط رسم تام، فإنه يكون لزاما علينا طلب المنهاج **أنشر Display** لكل نوع من الكائنات التي تدخل في رسمنا. على غرار ذلك، لقد حافظنا على توقيعات مختلف مناهج النشر لكل الكائنات المنتمية لنفس تسلسل **كائن بياني**: إنه الشرط قبل استخدام تعدد الصفات. إذن، يمكننا الآن استعمال كود الشكل:

```
method Drawing :: Display
{
    for every GraphicObject include
```

```
    {
    [Object Display]
    }
}
```

إستعمال تعدد الصفات في مجموعة

تعدد الصفات للمنهاج **أنشر** يضمن مناداة المنهاج المناسب للكائن المناسب. الآلية الداخلية لهذا الميكانيزم العجيب ترتكز على استراتيجية الربط المتأخر أو *late Bending*. لنفرض برنامجا كلاسيكيا، عنوان طلب إجراء أو دالة يُحسب أثناء تحرير الروابط، ويُشفّر بدقة في البرنامج: إنه الربط المتسرّع (*early Bending*). في حالة الربط المتأخر، مكان المنهاج المطلوب يقع في الكائن نفسه. إذن في أثناء التنفيذ ينشئ البرنامج عنوان الطلب.

قد يكون هذا المثال مبهما، فلنفرض مثالا آخر، ويتعلق الأمر بالمنهاج **تحرك إلى MoveTo**، والذي شرحناه سابقا. إذن هذا الكود يعد صالحا لأي نوع من أنواع الكائن البياني الذي نطبقه عليه: التحريك يعتمد دائما على محو متواصل، تغيير للإحداثيات ثم نشر. مرة أخرى، يعمل هذا الكود بفضل تعدد الصفات polymorphism لأنه مطلوب أن تكون مناهج المحو والنشر المناسبة للكائن المراد هي التي تم طلبها.

نفس الشيء، إذا اعتبرنا أن المحو يتلخص في إعادة رسم الكائن ولكن بلون الخلفية، فإنه يمكننا تعريف منهاج المحو بهذا الشكل:

```
method GraphicObject :: Delete
{
    [Object SetColor: BackgroundColor]
    [Object Display]
}
```

إستعمال تعدد الصفات في المنهاج delete

٥ . ٣ . شكل ناجع لتعدد الصفات : التحميل الزائد Overloading

التحميل الزائد هو آلية مقترحة بكثرة من قبل لغات التوجه الكائني، والذي يسمح بتخصيص توقيعات مختلفة لمناهج / دوال / إجراءات تحمل نفس الإسم.

كمثال، نقدر أن نقترح توقيعين مختلفين للمنهاج **أنشر**

- من دون بارامترات إذا ما أردنا استعمال وسيط نشر بالغياب Default.
- تحديد وسيط ببارامتر.

٦ علاقة الشراكة : The Relation of Association

الشراكة (البرمجية وليست الاقتصادية) هي ثالث نوع مهم من العلاقة التي فرضناها بعد كل من الوراثة Heritage والتركيب Aggregation، فإذا كانت الوراثة لا تعاني من أي

إلتباس، لأنها تترجم الجملة "...هي شكل خاص من ... (IS A)". فإن علاقة الشراكة يصعب تمثيلها والرمز إليها. على كل، فحسب مختلف الناشرين، فإنها يمكننا القول: "... يتصل مع ..." ، أو "... يستعمل ..."، أو بالإنجليزية " USES A ". ففي بعض الحالات، يمكن بسهولة أن تختلط الأمور مع علاقة التركيب aggregation ("... يتركب من ...") كما سنرى في المثال الذي سيأتي لاحقا.

من أجل تثبيت المفاهيم، فلنفرض مثالا كلاسيكيا ، ويتعلق الأمر بحديقة الحيوانات. فمن جهة نذكّر أن مبدأ الكبسلة شخصي، ويتجاوب مع تفكير المصمم.

حديقة الحيوانات إذن تتكون من:

- مجموع أقفاص
- مجموعة حيوانات
- مجموعة حراس

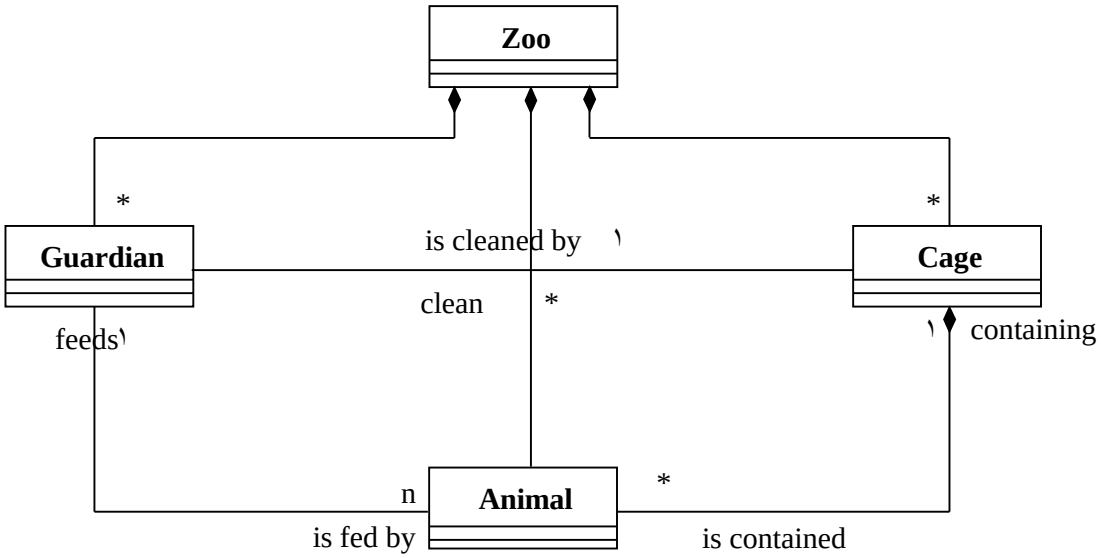
يظهر بوضوح أن هذه العلاقات من التركيب aggregation.

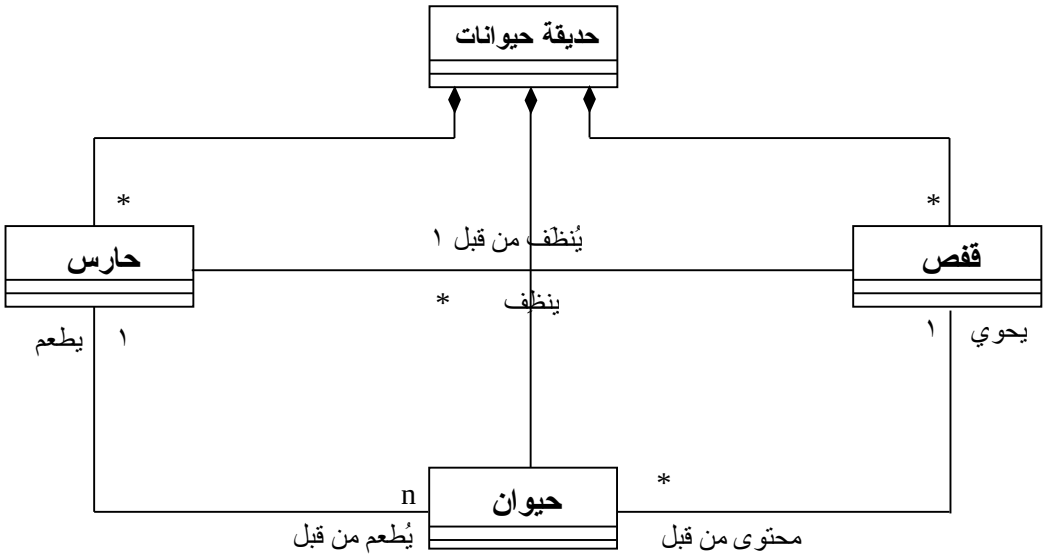
عوض ذلك، أي حارس يجب أن يراقب عددا معينا من الحيوانات (حسب قوانين إدارة الحديقة)، وينظف عددا آخر من الأقفاص. على نفس المنوال، أي قفص يحوي عددا معينا من الحيوانات (دائما حسب إدارة الحديقة).

العلاقات الأخيرة لا صلة لها بالتركيب aggregation (عادة ما نعتبر أن نفس الكائن ليس معتمدا من قبل الكائنات الأخرى) ولكنها شراكات. إذن نقوم بجعل كل علاقة لا تتناسب مع الوراثة أو التركيب ضمن حالة الشراكة، فنحصل إذن على التصميم الذي سيأتي لاحقا.

على طريقة التركيب aggregation، سوف نحدد كارديناليات ومهام على علاقة الشراكة. كمثال، إذا ما اعتبرنا العلاقة بين الصنفين **قفص** و **حارس**، نستطيع قراءة:

"أي حارس يُنظِف من ٠ إلى ن قفص / أي قفص يُنظَّف من قبل حارس واحد فقط"

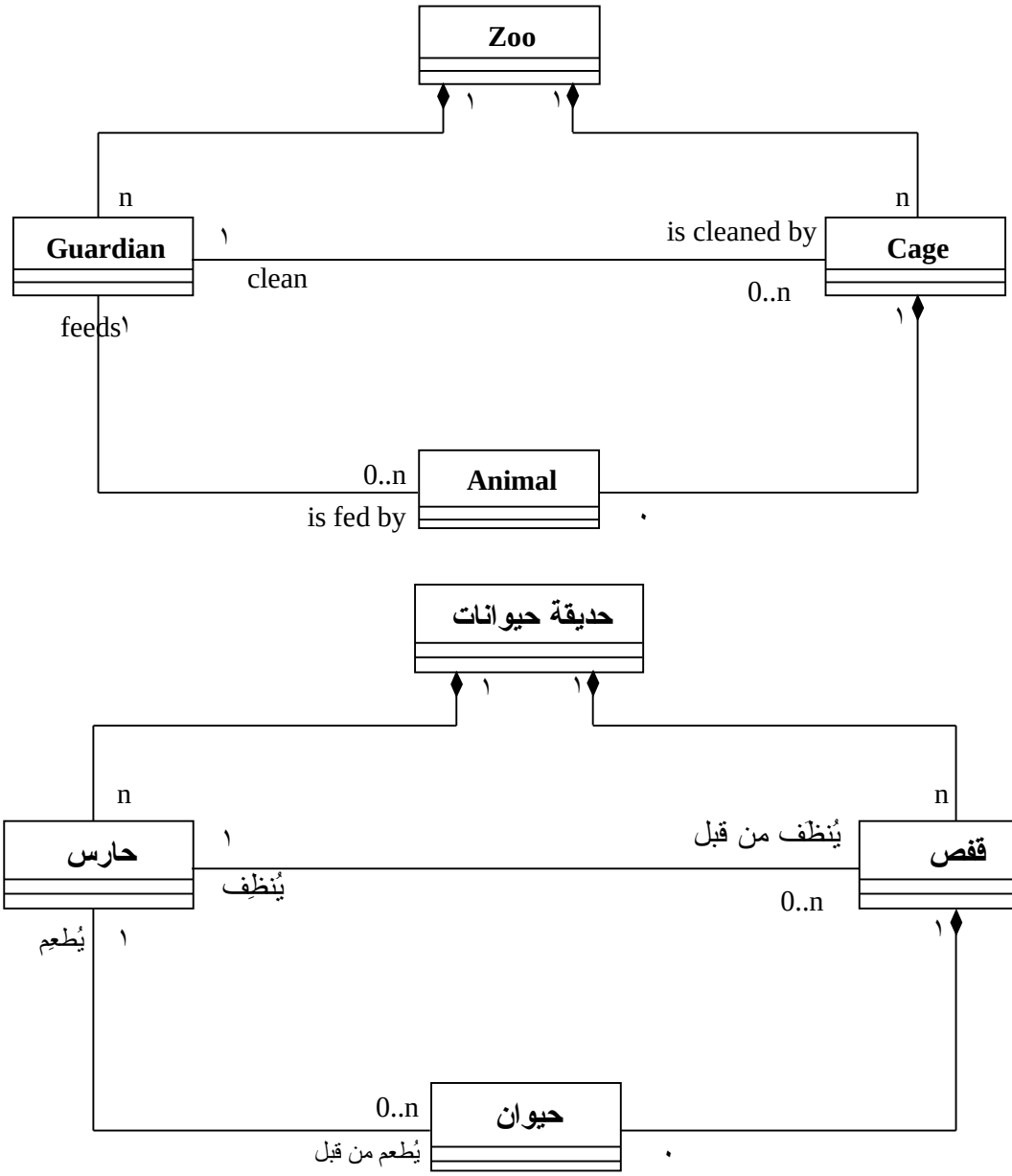




<u>قولبة حديقة حيوانات بالتركيب والشراكة</u>

رغم ذلك، غالبا ما يصعب تحديد ما هو <u>تركيب</u> مم هو <u>شراكة</u>. فكمثال ثاني، نموذج ما يحدد أن الحديقة تركب aggregate الحراس والأقفاص، وهذه الأخيرة تركب حيوانات.

هذه الحالة تصمم بهذا الشكل:



قالب آخر لحديقة حيوانات بالتركيب والشراكة

٧ ختام نموذج الكائن:

لقد رأينا دراسة مبسطة لجزء قليل من المفاهيم المتعلقة بمفهوم الكائن، ولكن المشوار مازال طويلا، لأن المفاهيم كثيرة ومميزاتها واسعة. علاقات الوراثة، التركيب والشراكة هي الأساسية في الكائنات. فبعض المبرمجين يرى أنه من خلال هذه العناصر الثلاثة يمكن فعل كل شيء، آخرون يتناولون أنواع أخرى من العلاقات، ويعملون بها. النقطة الأساسية الواجب حفظها هي أن القولبة تعتمد اعتمادا كليا على مفهوم الكبسلة والمرتبط بتفكير المصمم، وبحالة التطبيق المراد إنجازه.

قبل الختام، أشير إلى أن هذا العمل هو الأول من نوعه بالنسبة لي، ويعد بذلك اجتهادا شخصيا. قد ينقصه الكثير من المفاهيم أو التنظيم أو حتى طريقة الشرح، فلا لا تترددوا باقتراحاتكم وإضافاتكم، وشكرا.

٨ قاموس المفردات :

المصطلح الإنجليزي	المرادف العربي	الشرح
Object	كائن	هو مجموعة من الدوال والبيانات التي تعمل في إطار موحد مشكلة شيئا متناسقا.
Instance	مثيل	هو صورة منفصلة من الصور التي تم تشكيلها من صنف ما.
Instanciation	خلق مثيل	عملية تشكيل كائن أو كائنات عديدة من نفس الصنف.
Heritage	وراثة	عملية اشتقاق أصناف جديدة من أصناف أكثر عمومية لتحوي بذلك على مميزاتها، مع القدرة على إضافة مميزات جديدة وعلاقتها هي: **الصنف المشتق هو إصدار خاص من الصنف القاعدي**
Encapsulation	كبسلة	إخفاء بعض البيانات عن المستخدم لحماية الكائن من تعديلات غير مرغوبة.
Aggregation	تركيب	هو أسلوب يسمح بتحديد مركبات كل صنف (أو كائن) وعددها، أو بمعنى آخر، العلاقة بين الأصناف، لتركيب صنف جامع.
Polymorphism	تعدد الصفات	هو أسلوب يسمح لصنف ما من أن يتخذ عدة أشكال، ومع نفس الأسماء يمكن طلب مناهج مختلفة تحمل نفس الإسم، ولكن تختلف في تركيبها.
Class	صنف	هو قالب يتم من خلاله خلق كائنات متشابهة أو مختلفة.
Method	منهاج	هو دالة من الدوال التابعة لكائن ما، ومن بينها المشيد والمهدم.
Attributes	خصائص	مختلف البيانات التابعة لكائن ما، سواء المحمية أو العامة أو الخاصة.
Constructor	مشيّد	هو المنهاج الذي يقوم بتهيئة الخصائص والبيانات التابعة للكائن عند أول عملية لخلق الكائن.
Destructor	مهدّم	هو المنهاج الذي يقوم بتحطيم الكائن بعد نهاية العمل، لتحرير الذاكرة المحجوزة.
Overload	التحميل الزائد	يسمح للمناهج أو الدوال أن تكون لها أسماء متشابهة، ولكن ببارامترات مختلفة.
Interface	الواجهة	تمثل مختلف أقسام الصنف (عام، خاص، محمي، افتراضي...) أو بمفهوم آخر قائمة مناهج الصنف.
Modeling	قولبة	إجراء تصنيف للكائنات المتشابهة (في أي صفة) لاستخلاص عدة قوالب جامعة ومتسلسلة.
Page	ورقة	هو الصنف الذي لا يمتلك مشتقات
Association	الشراكة	علاقة تربط بين الأصناف ولا تكون محددة كالتركيب aggregation.